software **development** academy

GIT Basics

Tasks

Tasks for the GIT Basics section

Dear student!

We encourage you to do tasks related to the GIT section.

It will allow you to consolidate your knowledge and prepare better for further classes.

Good luck!

Task 1

Task: command line

Launch the git bash console. Then check with the appropriate command in what directoryyou currently are. Next, create a directory named **Test** in your current location and with the usage of the appropriate command get into this newly created directory. The next step will be to create a text file named **test_file.txt**. The last part of the task is to edit the created file using editor **VI** - the vi command will do this.

Hint: to enter the file using the VI editor, type in the console: **vi filename**, and to enter edit mode use the ins (INSERT) key.

Enter the text "This is just a test" into the file and save the file.

Hint: to exit the edit mode, use the ESC key, and in order to save the file, use the following command **:wq**

Note: all actions should be performed using only the git bash console.

Task 2

Task: Creating a local repository

Launch the git bash console. Then create a directory named **my-first-git-repository** in the root directory and navigate to it. With the appropriate command initialize the current directory as a local git repository. To which branch were we automatically switched to after repository initialization?

Task 3

Task: Undoing the state of a file

Go to the repository created in the previous task (or create a new one with the appropriate one command). Then create two files from the git bash: **test1.txt**, **test2.txt** - with the appropriate command add the file test1.txt to the staging area and then try to delete it (everything should bemade from the git bash command line).

In the next part of the task, add the test2.txt file to the staging area the same way as previously. Then modify it by adding a line of text inside of it. Use the appropriate command to undo the changes. Finally, try to delete the file.

Task 4

Task: Change history

In the existing repository create a new file called **first-commit.txt** - add it to the staging area. Then modify its content in any way. Try to commit the changes to the file using the appropriate command. In the next part of the task create another file named **next-commit.txt** - add it to the staging area then modify its contents, also, modify the **first-commit.txt** file and commit all these changes. Try to go back one commit back. Finally, display the history of changes in the repository (appropriate command in the git bash console).

Task 5

Task: Creating a remote repository

The task will be to create a remote repository in one of the most popular Git hostings - GitHub.

To do this, visit https://github.com/ . If you don't have an account there yet - create one. The next step will be setting up a new remote repository: go to the GitHub homepage and use the green button labeled **‘New’** - you should be redirected to the repository creation page (https://github.com/new). Here you have to fill in the field **‘Repository name’** (required) representing the name of the repository - let's denote it as public (checkbox Public) - confirm changes using the **‘Create repository**’ button. The second part of the task will be cloning the remote repository to our local machine - for this purpose, you have to go to our repository page on GitHub and use the **‘Clone or download’** button - this way we will copy the address of our repository. The copied address must then be used in the git bash console, e.g. **git clone copied_repository_address**. In this way, we have cloned a remote repository to our local computer.

Task 6

Task: gitignore file

In your local repository synchronized with the remote (e.g. Created on GitHub) add a .gitignore file. Also, create a directory called **excluded**, and in that repository create a file named **excluded_file.txt**. The task will be to modify the .gitignore file so that:

● the **excluded** directory and its contents are ignored

● all files with the **.bat** extension are ignored

Test the operation: try adding some files with the extension .bat. Commit all changes, then push them to a remote repository.

Task 7

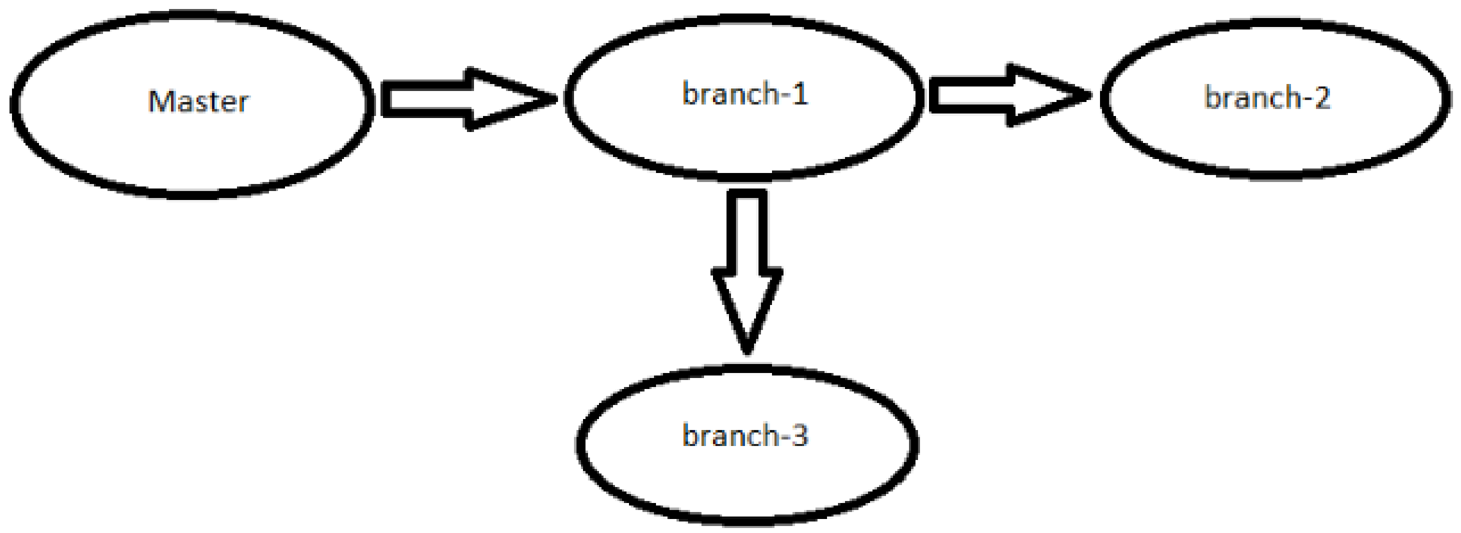
Task: Creating branches

In your local repository synchronized with the remote (created on GitHub) create a local branch named **branch-1** from a **master** branch. Then switch to the newly created branch. Create a new file called **branch-1-test.txt** - update it by adding any line of text. Commit introduced changes. Being on the **branch-1** branch, create on its basis another branch called **branch-2**. Switch to it and create two empty files named: **branch-2-test.txt** and **branch-2-test-2.txt**. Confirm the changes using the appropriate command. Then switch back to **branch-1** and create on its basis another branch called **branch-3** - similarly, now switch to the newly created branch, create an empty file **branch-3-test.txt** and commit changes.

Branches diagram below:

... the task is continued on the next slide

Task 7



Task 8

Task: Merging branches

Based on the previous task, where we created branches, we will now try to merge them together. Switch to branch **branch-3** and merge it with **branch-2** branch using the appropriate command in the git bash console. Then switch to branch **master** and merge it with the **branch-3** branch. Check with the appropriate command whether files **branch-1-test.txt, branch-2-test.txt, branch-2-test-2.txt**, **branch-3-test.txt** exist on the master branch.

Task 9

Task: Merge conflicts

Let's simulate the work of two people working on the same code, basing on the previous task about merging branches: Create a branch named **branch-conflict** from the **master** branch. Also, create another branch called **branch-conflict-2** from the **master** branch. Switch to **branch-conflict** branch and modify the **branch-1-test.txt** file - commit it - and then merge the **branch-conflict** branch to the master. Switch to **branch-conflict-2** and proceed as for the **branch-conflict** branch, that is: modify the same **branch-1-test.txt** file - commit it and try to do merge branch **branch-conflict-2** to master. What will happen? How to solve arisen problem?

Tip: You can use the available tool to resolve conflicts in IntelliJ Idea: position cursor over the file and select **Git-> Resolve Conflicts...** option from the context menu (right mouse button)

Task 10

Task: Rebase

Go to the repository that is synchronized with the remote one (on GitHub). Create two commits, in which some files will be modified or added (whichever you prefer). Then create an interactive rebase for two newly created commits - practice several different commands, which will be listed in the editor window (e.g. pick, reward, edit, fixup). Push changes (commits) to the remote repository.

What are your thoughts when using different rebase options?

Congratulations!

Congratulations on completing all tasks!

We encourage you to take the knowledge quiz and additional test.

Also please share your opinion on the video training in the survey!